

Efficient Reassembling of Graphs, Part 2: The Balanced Case

Saber Mirzaei *
Boston University

Assaf Kfoury *
Boston University

Abstract

The *reassembling of a simple connected graph* $G = (V, E)$ with $n = |V| \geq 1$ vertices is an abstraction of a problem arising in earlier studies of network analysis. The reassembling process has a simple formulation (there are several equivalent formulations) relative to a binary tree \mathcal{B} – its so-called *reassembling tree*, with root node at the top and n leaf nodes at the bottom – where every cross-section corresponds to a partition of V (a block in the partition is a node in the cross-section) such that:

- the bottom (or first) cross-section (*i.e.*, all the leaves) is the finest partition of V with n one-vertex blocks,
- the top (or last) cross-section (*i.e.*, the root) is the coarsest partition with a single block, the entire set V ,
- a node (or block) in an intermediate cross-section (or partition) is the result of merging its two children nodes (or blocks) in the cross-section (or partition) below it.

The *edge-boundary degree* of a block A of vertices is the number of edges with one endpoint in A and one endpoint in $(V - A)$. The **maximum** edge-boundary degree encountered during the reassembling process is what we call the **α -measure** of the reassembling, and the **sum** of all edge-boundary degrees is its **β -measure**. The α -optimization (resp. β -optimization) of the reassembling of G is to determine a reassembling tree \mathcal{B} that minimizes its α -measure (resp. β -measure).

There are different forms of reassembling, depending on the shape of the reassembling tree \mathcal{B} . In an earlier report, we studied *linear reassembling*, which is the case when the height of \mathcal{B} is $(n-1)$. In this report, we study *balanced reassembling*, when \mathcal{B} has height $\lceil \log n \rceil$. In a forthcoming report, we study *general reassembling*, which is the case when the height of \mathcal{B} can be any number between $(n-1)$ and $\lceil \log n \rceil$.

The two main results in this report are the NP-hardness of α -optimization and β -optimization of balanced reassembling. The first result is obtained by a sequence of polynomial-time reductions from *minimum bisection of graphs* (known to be NP-hard), and the second by a sequence of polynomial-time reductions from *clique cover of graphs* (known to be NP-hard).

1 Introduction

A more extensive introduction to the problem of graph reassembling, including the motivation for studying it, is in our earlier report [15]. We here limit ourselves to a brief review of the *balanced* case of the problem.

Problem Statement. Let $G = (V, E)$ be a simple (no self-loops, no multi-edges), connected, undirected graph, with $|V| = n \geq 1$ vertices and $|E| = m$ edges. The *reassembling* of G can be defined relative to a binary tree \mathcal{B} – one root node at the top, n leaf nodes at the bottom – where every cross-section corresponds to a partition of V (a block in the partition is a node in the cross-section) such that:

- the bottom (or first) cross-section (*i.e.*, all the leaves) is the finest partition of V with n one-vertex blocks,
- the top (or last) cross-section (*i.e.*, the root) is the coarsest partition with a single block, the entire set V ,
- a node (or block) in an intermediate cross-section (or partition) is the result of merging its two children nodes (or blocks) in the cross-section (or partition) below it.

*Partially supported by NSF awards CCF-0820138 and CNS-1135722.

For convenience, we say *vertices* to refer to the vertices of G and *nodes* to refer to those of the tree \mathcal{B} . We call G the *input graph* and \mathcal{B} the *reassembling tree*.

The height of \mathcal{B} is at least $\lceil \log n \rceil$ and at most $(n - 1)$. We say the reassembling of G according to \mathcal{B} is *balanced* if \mathcal{B} 's height is $\lceil \log n \rceil$; this is the case when, at every level of the reassembling, there is a maximum number of pairs of blocks which are each merged into a block at the next level up.

If A and B are disjoint nonempty sets of vertices in G , a *bridge* between A and B is an edge whose two endpoints are one in A and one in B . We write $\partial(A, B)$ to denote the set of all bridges between A and B . In case $B = V - A$, the set $\partial(A, B)$ is the same as the edge cut-set determined by the cut $(A, V - A)$, and instead of writing $\partial(A, V - A)$ we write $\partial(A)$ for simplicity. The *edge-boundary degree* of A is the number of bridges with only one endpoint in A , i.e., $|\partial(A)|$.

Various optimization problems can be associated with graph reassembling. Two such optimizations are the following, identified by the letters α and β throughout. We want to determine a reassembling tree \mathcal{B} for which:

- (α) the **maximum** edge-boundary degree encountered during reassembling is minimized, or
- (β) the **sum** of all edge-boundary degrees encountered during reassembling is minimized.

Initially, before we start reassembling, we always set the α -measure M_α to the **maximum** of all the vertex degrees, i.e., $\max \{degree(v) \mid v \in V\}$, and we set the β -measure M_β to the **sum** of the vertex degrees, i.e., $\sum \{degree(v) \mid v \in V\}$. During reassembling, after we merge disjoint nonempty subsets A and B , we update the α -measure M_α to: $\max \{M_\alpha, |\partial(A \cup B)|\}$, and the β -measure M_β to: $(M_\beta + |\partial(A \cup B)|)$. The reassembling process terminates when we reach the root of the reassembling tree \mathcal{B} .

About Terminology. Our choice of names comes from applications we studied in earlier report [15], where we had to disassemble and reassemble flow networks in such a way that the resulting α -measure and β -measure were minimized. However, *graph reassembling* as here defined can also be viewed as belonging to the large family of *graph embedding* problems, whereby a graph G (often called a *source graph* in the graph-theoretic literature) is embedded into another graph G' (often called a *host graph*) in such a way that various optimization measures (motivated by applications) are minimized or maximized.

For our reassembling problem, a source graph is what we call an *input graph* $G = (V, E)$, and a host graph is a rooted binary tree, which we call a *reassembling tree* \mathcal{B} . Moreover, the embedding in our problem takes a special form (namely, as explained above, there is a one-one correspondence between the vertices of G and the leaves of \mathcal{B} , the full set V is mapped to the root of \mathcal{B} and etc.).

We stick to our terminology, in part to be consistent with our earlier reports, where we study or use graph reassembling. But we also avoid in this report the use of concepts and terminology that are incidental to our graph-theoretic examination.

Main Results. We restrict attention to the balanced case of graph reassembling in this report. We prove that α -optimization and β -optimization of balanced reassembling are both NP-hard problems. We obtain these results by showing that:

- there is a sequence of several polynomial-time reductions from *minimum bisection of graphs* (MinBisection) to α -optimization of balanced reassembling of graphs,
- there is a sequence of several polynomial-time reductions from *clique cover of graphs* (CliqueCover) to β -optimization of balanced reassembling of graphs.

Both MinBisection and CliqueCover have been extensively studied: They are both NP-hard in general [8, 12]. This leaves open the problem of identifying classes of graphs, whether of practical or theoretical significance, for which there are low-degree polynomial-time solutions for our two optimization problems.

Organization of the Report. In Section 2 we give precise formal definitions of several notions underlying our examination of the balanced case. Section 3 presents the NP-hardness of α -optimization, and Section 4 the NP-hardness of β -optimization, for the balanced case of graph reassembling. We believe some of the intermediate reductions in Section 3 and Section 4 are of independent interest. Section 5 is a short wrap-up of related and future work.

2 Notational Conventions and Preliminary Definitions

We agree on notations and conventions for this report, and reproduce enough from the earlier [15] to make the present one self-contained.

All graphs are simple (no self-loops, no multi-edges) and undirected. With no loss of generality, we assume that all graphs have each an even number of vertices.

We denote the set of vertices of a graph G by $V(G)$, and its set of edges by $E(G)$. If there is an edge connecting two vertices $v, w \in V(G)$, we write \overline{vw} for the two-element set representing that edge. Unless explicitly stated otherwise, we reserve the letter “ n ” for the number of vertices and the letter “ m ” for the number of edges.

For convenience, we may refer to the graph G by writing $G = (V, E)$ instead of $G = (V(G), E(G))$, and also refer to the sizes of $V(G)$ and $E(G)$ by writing $|V|$ and $|E|$ instead of $|V(G)|$ and $|E(G)|$.

A graph H is a *subgraph* of the graph G iff $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. In this report we only need to consider subgraphs that are each induced by a subset of vertices. The subgraph H of G is said to be *induced* by a subset $A \subseteq V(G)$ iff $V(H) = A$ and $E(H) = \{\overline{vw} \in E(G) \mid v, w \in A\}$. We write $G[A]$ to denote the subgraph of G induced by the subset A of vertices.

We write $\partial_G(A)$ or $\partial(G[A])$ or, if G is clear from the context, just $\partial(A)$ to denote the *edge-boundary* of the subgraph $G[A]$, i.e., $\partial(A) = \{\overline{vw} \in E(G) \mid v \in A \text{ and } w \notin A\}$.

More generally, if A and B are disjoint subsets of $V(G)$, the set of *bridges* between A and B is denoted by $\partial_G(A, B)$ or, if G is clear from the context, just $\partial(A, B)$ which is the set $\{\overline{vw} \in E(G) \mid v \in A \text{ and } w \in B\}$. Thus, $\partial(A)$ is the same as $\partial(A, V - A)$.

We write $\text{GRAPHS}(2^*)$ to refer to the class of all simple graphs which have each a power-of-2 number of vertices.

Definition 1 (Minimum Bisection Problem). A *bisection* of the graph $G = (V, E)$ is a partition $\{A, B\}$ of $V(G)$ with two blocks of equal size:

$$|A| = |B|, \quad A \cap B = \emptyset, \quad \text{and} \quad A \cup B = V.$$

(Our running assumption is that V has an even number of vertices and can therefore be partitioned into two equal-size blocks.) We say the bisection $\{A, B\}$ is of *type* (X, Y) iff $X, Y \subseteq V(G)$ such that:

- either $X \cap Y = \emptyset$, $X \subseteq A$, and $Y \subseteq B$,
- or $X \cap Y = \emptyset$, $X \subseteq B$, and $Y \subseteq A$.

A *minimum bisection* $\{A, B\}$ of G is one that minimizes the set of bridges $\partial(A, B)$ between A and B , i.e.,

$$|\partial(A, B)| \leq \min \{ |\partial(A', B')| \mid \{A', B'\} \text{ is a bisection of } G \}.$$

The *minimum bisection problem*, MinBisection, asks for finding a minimum bisection of a given input graph G . MinBisection is known to be NP-hard [8].

We write $\text{MinBisection}(2^*)$ for the restriction of MinBisection to the class $\text{GRAPHS}(2^*)$. Lemma 11 asserts that $\text{MinBisection}(2^*)$ is also NP-hard. \square

Definition 2 (*Clique Cover Problem*). Given an integer $k \geq 1$, a k -clique cover of the graph $G = (V, E)$ is a partition of $V(G)$ into k disjoint subsets $\{V_1, \dots, V_k\}$ such that each of the induced subgraphs $G[V_1], \dots, G[V_k]$ is a complete graph (or a clique in G).

The k -clique cover problem, k -CliqueCover in shorthand, is a (yes,no) question that asks whether a graph G has a k -clique cover. k -CliqueCover is polynomial-time solvable for $k = 1, 2$, and is known to be NP-complete for every $k \geq 3$ [12]. \square

Our notion of a *reassembling* of graph $G = (V, E)$ presupposes the notion of a *binary tree* over the finite set $V(G) = \{v_1, \dots, v_n\}$. Our definition of *binary trees* is not standard, but is more convenient for our purposes.¹

Definition 3 (*Binary trees*). An (*unordered*) *binary tree* \mathcal{B} over $V(G) = \{v_1, \dots, v_n\}$ is a collection of non-empty subsets of $V(G)$ satisfying three conditions:

1. For every $v \in V(G)$, the singleton set $\{v\}$ is in \mathcal{B} .
2. The full set $V(G)$ is in \mathcal{B} .
3. For every $X \in \mathcal{B}$, there is a unique $Y \in \mathcal{B}$ such that: $X \cap Y = \emptyset$ and $(X \cup Y) \in \mathcal{B}$.

The *leaf nodes* of \mathcal{B} are the singleton sets $\{v\}$, and the *root node* of \mathcal{B} is the full set $V(G)$. Depending on the context, we may refer to the members of \mathcal{B} as its *nodes* or as its *clusters*. Several expected properties of \mathcal{B} , reproducing familiar ones of a standard definition, are stated in the next two propositions. \square

Proofs for Propositions 4 and 5 are straightforward and here omitted. Details can be found in the earlier report [15].

Proposition 4 (Properties of binary trees). *Let \mathcal{B} be a binary tree as in Definition 3, let $v \in V(G)$, and let:*

$$(\dagger) \quad \{v\} = X_0 \subsetneq X_1 \subsetneq X_2 \subsetneq \dots \subsetneq X_p = V(G)$$

be a maximal sequence of nested clusters from \mathcal{B} . We then have:

1. *The sequence in (\dagger) is uniquely defined, i.e., every maximal nested sequence starting from $\{v\}$ is the same.*
2. *For every cluster $Y \in \mathcal{B}$, if $\{v\} \subseteq Y$, then $Y \in \{X_0, \dots, X_p\}$.*
3. *There are pairwise disjoint clusters $\{Y_0, \dots, Y_{p-1}\} \subseteq \mathcal{B}$ such that, for every $0 \leq i < p$:*

$$X_i \cap Y_i = \emptyset \quad \text{and} \quad X_i \cup Y_i = X_{i+1}.$$

Based on this proposition, we use the following terminology:

- We call the sequence in (\dagger) , which is unique by part 1, the *path* from leaf node $\{v\}$ to root node $V(G)$.
- Every cluster containing v is one of the nodes along this unique path, according to part 2.
- In part 3, Y_i is the *sibling node* of X_i , and both are the *children nodes* of the *parent node* X_{i+1} .

Proposition 5 (Properties of binary trees). *Let \mathcal{B} be a binary tree as in Definition 3. We then have:*

1. *For all clusters $X, Y \in \mathcal{B}$, if $X \cap Y \neq \emptyset$ then $X \subseteq Y$ or $Y \subseteq X$.*
2. *For every cluster $X \in \mathcal{B}$, the sub-collection of clusters $\mathcal{B}_X := \{Y \in \mathcal{B} \mid Y \subseteq X\}$ is a binary tree over X , with root node X .*
3. *\mathcal{B} is a collection of $(2n - 1)$ nodes/clusters.*

¹A standard definition of a binary tree T makes T a subset of the set of finite binary strings $\{0, 1\}^*$ such that:

- T is prefix-closed, i.e., if $t \in T$ and u is a prefix of t , then $u \in T$.
- Every node has two children, i.e., $t0 \in T$ iff $t1 \in T$ for every $t \in \{0, 1\}^*$.

The *root node* of T is the empty string ε , and a *leaf node* of T is a string $t \in T$ without children, i.e., both $t0 \notin T$ and $t1 \notin T$.

Let θ be a map from $V(G)$ to $V(G)$. We extend θ to a map on every subset $X \subseteq V(G)$ by defining: $\theta(X) := \{\theta(v) \mid v \in X\}$, and on every set of subsets $\mathcal{X} \subseteq \mathcal{P}(V(G))$ by defining $\theta(\mathcal{X}) := \{\theta(X) \mid X \in \mathcal{X}\}$. If θ is a bijection on $V(G)$, then θ is also a bijection on $\mathcal{P}(V(G))$.

Proposition 6 (Properties of binary trees). *Let \mathcal{B} be a binary tree as in Definition 3. If $\theta : V(G) \rightarrow V(G)$ is a bijection on $V(G)$, then:*

1. $\theta(\mathcal{B})$ is a binary tree, i.e., the collection of subsets in $\theta(\mathcal{B})$ satisfies the three conditions in Definition 3.
2. \mathcal{B} and $\theta(\mathcal{B})$ are isomorphic trees, i.e.,
 - θ maps the leaf nodes and the root node of \mathcal{B} to the leaf nodes and the root node of $\theta(\mathcal{B})$.
 - θ maps a path $\{v\} = X_0 \subsetneq X_1 \subsetneq \dots \subsetneq X_p = V(G)$ in \mathcal{B} to a path $\{\theta(v)\} = \theta(X_0) \subsetneq \theta(X_1) \subsetneq \dots \subsetneq \theta(X_p) = V(G)$ in $\theta(\mathcal{B})$.
 - θ maps a pair of sibling nodes X and Y in \mathcal{B} to a pair of sibling nodes $\theta(X)$ and $\theta(Y)$ in $\theta(\mathcal{B})$.

In words, every bijection on $V(G)$ lifts to an isomorphism between binary trees on $V(G)$.

Proof. Straightforward from Definition 3. □

A common measure for a standard definition of binary trees is *height*, which is represented in the following, corresponding to our notion of binary trees in Definition 3:

$$\text{height}(\mathcal{B}) := \max \left\{ p \mid \text{there is } v \in V(G) \text{ such that } \{v\} = X_0 \subsetneq X_1 \subsetneq \dots \subsetneq X_p = V(G) \text{ is a maximal sequence of nested clusters} \right\}.$$

For a particular node/cluster $X \in \mathcal{B}$, the subtree of \mathcal{B} rooted at X is \mathcal{B}_X , by part 2 in Proposition 5. The height of X in \mathcal{B} is therefore $\text{height}_{\mathcal{B}}(X) := \text{height}(\mathcal{B}_X)$.

The binary tree \mathcal{B} over $\{v_1, \dots, v_n\}$ is *balanced* if $\text{height}(\mathcal{B}) = \lceil \log n \rceil$. If n is a power of 2, say $n = 2^p$, then $\text{height}(\mathcal{B}) = p$.

Definition 7 (Graph Reassembling). A *reassembling* of the graph $G = (V, E)$ is simply defined by a pair (G, \mathcal{B}) where \mathcal{B} is a binary tree over $V(G)$, as in Definition 3.

Given a reassembling (G, \mathcal{B}) of G , two measures are of particular interest for our analysis, namely, for every node/cluster $X \in \mathcal{B}$, the *edge-boundary degree* (or simply the *degree*) of X and the *height* of X in \mathcal{B} :

$$\text{degree}_{G, \mathcal{B}}(X) := |\partial_G(X)| \quad \text{and} \quad \text{height}_{G, \mathcal{B}}(X) := \text{height}_{\mathcal{B}}(X).$$

If the context makes clear the reassembling (G, \mathcal{B}) – respectively, the binary tree \mathcal{B} – relative to which of these measures are defined, we write $\text{degree}(X)$ and $\text{height}(X)$ – or $\text{degree}_G(X)$ and $\text{height}_G(X)$, respectively – instead of $\text{degree}_{G, \mathcal{B}}(X)$ and $\text{height}_{G, \mathcal{B}}(X)$.²

We say the reassembling (G, \mathcal{B}) is *balanced* if the underlying binary tree \mathcal{B} is balanced. □

Definition 8 (Isomorphic Graph Reassemblings). Let (G, \mathcal{B}) and (G, \mathcal{B}') be two reassemblings of the same graph G . We say (G, \mathcal{B}) and (G, \mathcal{B}') are *isomorphic reassemblings* if there is a bijection $\theta : V(G) \rightarrow V(G)$ satisfying two conditions:

1. $\mathcal{B}' = \theta(\mathcal{B})$.
2. For every node $X \in \mathcal{B}$, we have $\text{degree}_{G, \mathcal{B}}(X) = \text{degree}_{G, \mathcal{B}'}(\theta(X))$.

In words, for (G, \mathcal{B}) and (G, \mathcal{B}') to be isomorphic, not only do we require that the reassembling trees \mathcal{B} and \mathcal{B}' be isomorphic (Proposition 6), but also that the degrees of every node X and its image $\theta(X)$ be equal. □

² Our reassembling of G can be viewed as “hierarchical clustering” of G , similar to a method of analysis in data mining, though used for a different purpose. Our reassembling mimicks so-called “agglomerative, or bottom-up, hierarchical clustering” in data mining.

The following definition repeats a definition in Section 1 more formally.

Definition 9 (*Measures on the reassembling of a graph*). Let (G, \mathcal{B}) be a reassembling of G . We define the measures α and β on (G, \mathcal{B}) as follows:

$$\begin{aligned}\alpha(G, \mathcal{B}) &:= \max \{ \text{degree}_{G, \mathcal{B}}(X) \mid X \in \mathcal{B} \}, \\ \beta(G, \mathcal{B}) &:= \sum \{ \text{degree}_{G, \mathcal{B}}(X) \mid X \in \mathcal{B} \}.\end{aligned}$$

An optimization problem arises with the minimization of each of these measures. We say the reassembling (G, \mathcal{B}) is α -optimal iff:

$$\alpha(G, \mathcal{B}) = \min \{ \alpha(G, \mathcal{B}') \mid \mathcal{B}' \text{ is a binary tree over } V(G) \}.$$

We say (G, \mathcal{B}) is an α -optimal balanced reassembling iff:

$$\alpha(G, \mathcal{B}) = \min \{ \alpha(G, \mathcal{B}') \mid \mathcal{B}' \text{ is a balanced binary tree over } V(G) \}.$$

Important note: When we say “ (G, \mathcal{B}) is an α -optimal balanced reassembling,” we mean (G, \mathcal{B}) is α -optimal among all *balanced* reassemblings only, we do not mean that (G, \mathcal{B}) is α -optimal among all reassemblings and that (G, \mathcal{B}) happens to be balanced.

We leave to the reader the obvious similar definition of what it means for (G, \mathcal{B}) to be a β -optimal balanced reassembling. \square

3 α -Optimization of Balanced Reassembling Is NP-Hard

Consider an arbitrary graph $G = (V, E)$. Let p be the smallest integer such that $2^p \geq |V| = n \geq 2$. It follows that $n + r = 2^p$ for some even integer r such that $0 \leq r < n$. Another way of identifying r is to say it is the smallest even integer such $n + r$ is a power of 2. Our standing assumption is that n is even.

Let $q = (n/2) + r$. Following standard notation, K_q denotes the complete graph over q vertices. For Lemma 10, we construct an augmented graph \mathbb{G} consisting of the original G together with two disjoint copies of K_q , which we denote by the separate letters H and I for clarity. More precisely,

$$\begin{aligned}V(\mathbb{G}) &= V(G) \uplus V(H) \uplus V(I), \\ E(\mathbb{G}) &= E(G) \uplus E(H) \uplus E(I) \uplus \{ \overline{vw} \mid v \in V(G) \text{ and } w \in V(H) \cup V(I) \},\end{aligned}$$

where we write “ \uplus ” for “disjoint union”. We thus assemble the new \mathbb{G} by connecting every vertex in G with all the vertices in H and I . There are no edges between H and I . There is a total of $n + q + q = 2n + 2r = 2^{p+1}$ vertices in \mathbb{G} . Thus, the graph \mathbb{G} has between $2n$ and $3n$ vertices and is a member of the class $\text{GRAPHS}(2^*)$.

Lemma 10. *Consider the graph \mathbb{G} as defined in the preceding paragraph. Every minimum bisection of \mathbb{G} must be of type $(V(H), V(I))$ – see Definition 1 – schematically shown in Figure 1a.*

Proof. We consider a bisection $\{A, B\}$ of \mathbb{G} which is *not* of type $(V(H), V(I))$, and then show that it cannot be a minimum bisection.

Because $\{A, B\}$ is a bisection, we have $|A| = |B| = n + r$. And because $|V(H)| = |V(I)| = (n/2) + r$, it is never the case that $A \cap (V(H) \uplus V(I)) = \emptyset$ or $B \cap (V(H) \uplus V(I)) = \emptyset$. Hence, there are two possible cases, one shown in Figure 1b and one in Figure 1c. In Figure 1b, the vertices of both H and I appear on both sides of the bisection. In Figure 1c, the vertices of H are all on the same side of the bisection, while the vertices of

I appear on both sides of the bisection. In these figures, $\{X, X', Y, Y', Z, Z'\}$ is a 6-block partition of $V(\mathbb{G})$, defined by:

$$\begin{aligned} X &= A \cap V(G) & \text{and} & & X' &= B \cap V(G) & (\text{the vertices of subgraph } G), \\ Y &= A \cap V(H) & \text{and} & & Y' &= B \cap V(H) & (\text{the vertices of subgraph } H), \\ Z &= A \cap V(I) & \text{and} & & Z' &= B \cap V(I) & (\text{the vertices of subgraph } I). \end{aligned}$$

In each of the two cases shown in Figures 1b and 1c, we need to prove that $|\partial(A, B)|$ can be decreased by moving an appropriate number of vertices of the subgraphs H and I from one side of the bisection to the other side.

Case 1. This is the case in Figure 1b. With no loss of generality, suppose:

$$|X| \geq |X'|.$$

Because $|A| = |B|$, this forces the inequality:

$$|Y| + |Z| \leq |Y'| + |Z'|.$$

Because $|Y| + |Y'| = |Z| + |Z'|$, this in turn forces one or both of the following inequalities:

$$(\dagger) \quad |Y| \leq |Z'| \quad \text{or} \quad |Z| \leq |Y'|.$$

With no loss of generality, assume the first inequality $|Y| \leq |Z'|$ in (\dagger) holds.

Let $|Y| = k \geq 1$. Select an arbitrary subset $U \subseteq Z'$ such that $|U| = k$. We define a new bisection $\{\tilde{A}, \tilde{B}\}$ of \mathbb{G} by moving: (1) all the vertices of Y from the A -side to the B -side, and (2) all the vertices of U from the B -side to the A -side. Specifically, let:

$$\tilde{A} = (A - Y) \cup U \quad \text{and} \quad \tilde{B} = (B - U) \cup Y.$$

The resulting set of edges connecting \tilde{A} and \tilde{B} is:

$$\begin{aligned} \partial(\tilde{A}, \tilde{B}) &= \left(\partial(A, B) - \left(\partial(Y, X') \cup \partial(Y, Y') \cup \partial(U, X) \cup \partial(U, Z) \right) \right) \\ &\quad \cup \left(\partial(Y, X) \cup \partial(U, X') \cup \partial(U, Z' - U) \right). \end{aligned}$$

Because $|\partial(U, X)| = |\partial(Y, X)|$ and $|\partial(U, X')| = |\partial(Y, X')|$, it follows:

$$|\partial(\tilde{A}, \tilde{B})| = |\partial(A, B)| - |\partial(Y, Y')| - |\partial(U, Z)| + |\partial(U, Z' - U)|$$

With the fact that $|Y| = |U|$, the following inequality must hold:

$$|\partial(Y, Y')| \geq |\partial(U, Z' - U)|$$

otherwise, if it did not, we would have that $|Y'| < |Z' - U| = |Z'| - |U| = |Z'| - |Y|$, in turn implying that $|Y| + |Y'| < |Z'|$, which is a contradiction. Hence,

$$|\partial(\tilde{A}, \tilde{B})| \geq |\partial(A, B)| - |\partial(U, Z)| > |\partial(A, B)|.$$

We conclude that $|\partial(\tilde{A}, \tilde{B})| < |\partial(A, B)|$.

Case 2. This is the case in Figure 1c. It cannot be that $|X| < |X'|$, because if it were so, it would imply $|X| < (n/2)$ which, with the fact that $|Z \cup Z'| = (n/2) + r$, would in turn imply that $|A| < n + r$, thus contradicting the hypothesis that $\{A, B\}$ is a bisection of \mathbb{G} . Hence, it must be that:

$$|X| \geq |X'| \geq n/2.$$

The largest possible size of X , which is $n - 1$, corresponds to the smallest possible size of Z which, because $|X| + |Z| = n + r$, must therefore be $r + 1$. Corresponding to the smallest possible size of Z is the largest possible size of Z' , which is therefore $|V(I)| - (r + 1) = (n/2) - 1$. Hence, it is always the case that $|Z'| < |X|$.

We now proceed in a way similar to **Case 1**. Let $|Z'| = k \geq 1$. Select an arbitrary subset $U \subseteq X$ such that $|U| = k$ and $\partial(U, X') \neq \emptyset$. The new bisection $\{\tilde{A}, \tilde{B}\}$ of \mathbb{G} is obtained by moving: (1) all the vertices of Z' from the B -side to the A -side, and (2) all the vertices of U from the A -side to the B -side. Specifically, let:

$$\tilde{A} = (A - U) \cup Z' \quad \text{and} \quad \tilde{B} = (B - Z') \cup U.$$

The resulting set of edges connecting \tilde{A} and \tilde{B} is:

$$\begin{aligned} \partial(\tilde{A}, \tilde{B}) = & \left(\partial(A, B) - \left(\partial(Z', Z) \cup \partial(Z', X - U) \cup \partial(Y', U) \cup \partial(X', U) \right) \right) \\ & \cup \left(\partial(Z', X') \cup \partial(Z', U) \cup \partial(Z, U) \cup \partial(X - U, U) \right). \end{aligned}$$

Because $|\partial(Y', U)| = |\partial(Z \cup Z', U)| = |\partial(Z, U)| + |\partial(Z', U)|$, it follows that:

$$\begin{aligned} |\partial(\tilde{A}, \tilde{B})| = & |\partial(A, B)| - |\partial(Z', Z)| - |\partial(Z', X - U)| - |\partial(X', U)| \\ & + |\partial(Z', X')| + |\partial(X - U, U)|. \end{aligned}$$

Because $|\partial(Z', X - U)| \geq |\partial(X - U, U)|$, we obtain the inequality:

$$|\partial(\tilde{A}, \tilde{B})| \leq |\partial(A, B)| - |\partial(Z', Z)| - |\partial(X', U)| + |\partial(Z', X')|.$$

Because $|\partial(Z', Z)| \geq |\partial(Z', X')|$, it follows that:

$$|\partial(\tilde{A}, \tilde{B})| \leq |\partial(A, B)| - |\partial(X', U)|.$$

Because $|\partial(X', U)| \neq 0$, we conclude that $|\partial(\tilde{A}, \tilde{B})| < |\partial(A, B)|$. □

Lemma 11. *MinBisection(2^*) is an NP-hard problem.*

Proof. We use the same notation as in the proof of Lemma 10. Given that subgraphs H and I of \mathbb{G} have an equal number $q = (n/2) + r$ of vertices, Lemma 10 implies we can reduce, in polynomial time, MinBisection for an arbitrary graph G to MinBisection(2^*) for graph \mathbb{G} . Hence, a minimum bisection for \mathbb{G} induces a minimum bisection for the given G . Hence, the NP-hardness of MinBisection in general implies the NP-hardness of MinBisection(2^*). □

Lemma 12. *Let $G = (V, E)$ be a graph in the class $\text{GRAPHS}(2^*)$ and let \mathbb{G} be the augmented graph of G as in Lemma 10. Let $(\mathbb{G}, \mathcal{B})$ be a balanced reassembling of \mathbb{G} where \mathcal{B} is a binary tree over $V(\mathbb{G})$ (see Definition 3). Let A and B be the two children nodes of the root node $V(\mathbb{G})$ in \mathcal{B} .*

Conclusion: *If $(\mathbb{G}, \mathcal{B})$ is an α -optimal balanced reassembling, then $\{A, B\}$ is a minimum bisection of \mathbb{G} .*

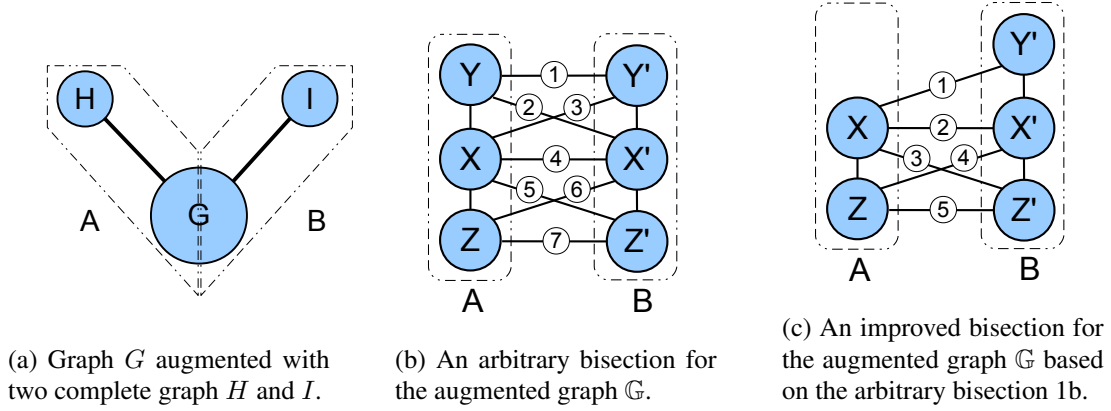


Figure 1: For the statement of Lemma 10.

Proof. Because $G \in \text{GRAPHS}(2^*)$, each of the complete graphs H and I has $(n/2)$ vertices, i.e., in contrast to the proof of Lemma 10, here $r = 0$. The augmented graph \mathbb{G} is also in the class $\text{GRAPHS}(2^*)$, with $|V(G)| = n$ and $|V(\mathbb{G})| = 2n$. In the given balanced reassembling $(\mathbb{G}, \mathcal{B})$, we have that $\{A, B\}$ is a bisection of \mathbb{G} , with $|A| = |B| = n$. The subtrees \mathcal{B}_A and \mathcal{B}_B rooted at A and B are each over n vertices. By Definitions 7 and 9, we have $\alpha(\mathbb{G}, \mathcal{B}) \geq |\partial(A, B)|$. The conclusion of the lemma will follow from the fact that $\alpha(\mathbb{G}, \mathcal{B}) = |\partial(A, B)|$, which we show next.

In the rest of the proof we use the notation and definitions in the proof of Lemma 10. There are two cases, depending on whether $\{A, B\}$ is, or is not, of type $(V(H), V(I))$.

Case 1. If $\{A, B\}$ is of type $(V(H), V(I))$, we can assume that $V(H) \subseteq A$ and $V(I) \subseteq B$. We pose:

$$X = A \cap V(G), \quad X' = B \cap V(G), \quad Y = A \cap V(H), \quad \text{and} \quad Z = B \cap V(I).$$

By construction, $|X| = |X'| = (n/2)$ and $|Y| = |Z| = (n/2)$. Let $C = |\partial(X, X')|$, which is the value of the bisection of the given graph G . Because H and I are copies of the complete graph $K_{(n/2)}$, the set of edges connecting Y to X' , and the set of edges connecting Z to X , satisfy the equalities:

$$|\partial(Y, X')| = |\partial(Z, X)| = (n/2)^2 = n^2/4.$$

Hence, using the notation of Definitions 7 and 9, we have:

$$|\partial(A, B)| = \text{degree}_{\mathbb{G}, \mathcal{B}}(A) = \text{degree}_{\mathbb{G}, \mathcal{B}}(B) = (n^2/4) + (n^2/4) + C = (n^2/2) + C.$$

Hence, $\alpha(\mathbb{G}, \mathcal{B}) \geq (n^2/2) + C$. The equality in fact holds because, as argued next, $\text{degree}_{\mathbb{G}, \mathcal{B}}(S) \leq (n^2/2) + C$ for every node/cluster of vertices S in the subtrees \mathcal{B}_A and \mathcal{B}_B .

Let S be a node in \mathcal{B}_A . (The same argument applies if S is a node in \mathcal{B}_B .) If $S = A$, we already know that $\text{degree}_{\mathbb{G}, \mathcal{B}}(S) \leq (n^2/2) + C$. Suppose S is not the root of \mathcal{B}_A , i.e., $S \neq A$. This implies $|S| \leq |A|/2$. Let $|S| = k$, so that $k \leq (n/2)$. Let $S_1 = S \cap X$ and $S_2 = S \cap Y$, with $|S_1| = k_1$ and $|S_2| = k_2$, so that also $k_1 + k_2 = k \leq (n/2)$. Note that $\partial(S_1, X') \subseteq \partial(X, X')$, so that if $C_1 = |\partial(S_1, X')|$, then $C_1 \leq C$. Also, $\partial(S_2, X') = \partial(S_2, Z) = \emptyset$. We conclude:

$$\text{degree}_{\mathbb{G}, \mathcal{B}}(S) = \underbrace{k_1(n/2) + C_1}_{|\partial(S_1, B)|} + \underbrace{k_1 k_2}_{|\partial(S_1, S_2)|} + \underbrace{k_2(n/2)}_{|\partial(S_2, X')|} = (k_1 + k_2)(n/2) + k_1 k_2 + C_1 \leq (n^2/2) + C.$$

Case 2. Suppose $\{A, B\}$ is not of type $(V(H), V(I))$. By Lemma 10, $\{A, B\}$ is not a minimum bisection and therefore $|\partial(A, B)| > (n^2/2) + C$ where C is defined as in Case 1 above. Hence $\alpha(\mathbb{G}, \mathcal{B}) > (n^2/2) + C$. By Case 1, $(\mathbb{G}, \mathcal{B})$ is not an α -optimal balanced reassembling. \square

Theorem 13. *For the class of simple undirected graphs G , the computation of α -optimal balanced reassemblies (G, \mathcal{B}) is an NP-hard problem.*

Proof. If a deterministic polynomial-time algorithm \mathcal{A} existed for computing an α -optimal balanced reassembling for an arbitrary graph G , then \mathcal{A} could be used for computing an α -optimal balanced reassembling for the augmented graph \mathbb{G} of an arbitrary graph $G \in \text{GRAPHS}(2^*)$. Hence, by Lemma 12, \mathcal{A} could also be used for computing a minimum bisection of \mathbb{G} in deterministic polynomial time. This would contradict the NP-hardness of $\text{MinBisection}(2^*)$, as asserted by Lemma 11. The desired conclusion follows. \square

4 β -Optimization of Balanced Reassembling Is NP-Hard

We need to introduce two variations of the k -CliqueCover problem (Definition 2).

Definition 14 (*Fixed-Size 4-CliqueCover, Equal-Size 4-CliqueCover*). In the *Fixed-Size 4-CliqueCover* problem we consider a graph G together with four positive integers $\{n_1, n_2, n_3, n_4\}$ such that $n_1 + n_2 + n_3 + n_4 = |V(G)|$ and we ask: Can we partition $V(G)$ into four disjoint subsets A_1, A_2, A_3 and A_4 of respective sizes n_1, n_2, n_3 and n_4 such that each of the induced subgraphs $G[A_1]$, $G[A_2]$, $G[A_3]$ and $G[A_4]$ is a complete graph (i.e., A_1, A_2, A_3 and A_4 are cliques)?

In the *Equal-Size 4-CliqueCover* problem we consider a graph G and ask: Can we partition $V(G)$ into four disjoint subsets A_1, A_2, A_3 and A_4 of equal size, i.e., $|A_1| = |A_2| = |A_3| = |A_4| = |V(G)|/4$, such that each of $G[A_1]$, $G[A_2]$, $G[A_3]$ and $G[A_4]$ is a complete graph (i.e., A_1, A_2, A_3 and A_4 are cliques)? \square

Lemma 15. *Fixed-Size 4-CliqueCover is NP-complete.*

Proof. Given a 4-part partition $\{A_1, A_2, A_3, A_4\}$ of $V(G)$, we can verify in polynomial time that the induced graphs $\{G[A_1], G[A_2], G[A_3], G[A_4]\}$ are each complete and that their sizes are the given $\{n_1, n_2, n_3, n_4\}$. So the problem is in NP.

We next show that NP-completeness follows by reduction from 4-CliqueCover, i.e., the existence of a deterministic polynomial-time algorithm \mathcal{A} for Fixed-Size 4-CliqueCover would imply the existence of a deterministic polynomial-time algorithm for 4-CliqueCover. The input for the hypothetical \mathcal{A} consists of a graph G together with four positive integers $\{n_1, n_2, n_3, n_4\}$. For the desired reduction, we use the function $p(n, 4)$, a cubic polynomial in n , which counts the number of ways of partitioning positive integer n into 4 positive integers [2]:

$$p(n, 4) := \begin{cases} \left\lceil \frac{(n+1)^3}{144} - \frac{(n+1)}{48} \right\rceil & \text{if } n \text{ is even,} \\ \left\lceil \frac{(n+1)^3}{144} - \frac{(n+1)}{12} \right\rceil & \text{if } n \text{ is odd,} \end{cases}$$

where $\lceil x \rceil$ is the nearest integer to x . We leave to the reader the straightforward task of writing an algorithm \mathcal{A}' to generate all partitions of n into 4 positive integers, which runs in $\mathcal{O}(n^3)$ time. To decide whether an arbitrarily given graph G is a positive instance of 4-CliqueCover, we run algorithm \mathcal{A}' to generate the successive 4-part partitions of $n = |V(G)|$. The given G has a 4-clique cover iff algorithm \mathcal{A} returns “yes” when its input is: G together with at least one of these 4-part partitions of integer n . \square

Lemma 16. *Equal-Size 4-CliqueCover is NP-complete.*

Proof. If $\{A_1, A_2, A_3, A_4\}$ is a 4-part partition of $V(G)$, where $|A_1| = |A_2| = |A_3| = |A_4| = |V(G)|/4 = n/4$, we can verify in polynomial time that the induced graphs $\{G[A_1], G[A_2], G[A_3], G[A_4]\}$ are each complete. So the problem is in NP.

NP-completeness follows by reduction from Fixed-Size 4-CliqueCover to Equal-Size 4-CliqueCover, *i.e.*, the existence of a deterministic polynomial-time algorithm \mathcal{A} for Equal-Size 4-CliqueCover would imply the existence of a deterministic polynomial-time algorithm for Fixed-Size 4-CliqueCover, as shown next. Given an arbitrarily given graph G and four positive integers $\{n_1, n_2, n_3, n_4\}$ such that $n_1 + n_2 + n_3 + n_4 = |V(G)| = n$, we introduce 4 new sets of vertices $\{A_1, A_2, A_3, A_4\}$ such that:

$$|A_1| = n - n_1, \quad |A_2| = n - n_2, \quad |A_3| = n - n_3, \quad |A_4| = n - n_4.$$

We construct a new graph G' such that:

$$\begin{aligned} V(G') &:= V(G) \cup A_1 \cup A_2 \cup A_3 \cup A_4, \\ E(G') &:= E(G) \cup \{\overline{vw} \mid v \in V(G) \text{ and } w \in A_1 \cup A_2 \cup A_3 \cup A_4\} \\ &\quad \cup \{\overline{vw} \mid v, w \in A_1\} \cup \{\overline{vw} \mid v, w \in A_2\} \cup \{\overline{vw} \mid v, w \in A_3\} \cup \{\overline{vw} \mid v, w \in A_4\}. \end{aligned}$$

In words, the new G' is obtained from G by adding four cliques, one clique on each of the new vertex sets in $\{A_1, A_2, A_3, A_4\}$, and by connecting every vertex in $A_1 \cup A_2 \cup A_3 \cup A_4$ with every vertex in $V(G)$. Hence, G' has $4n$ vertices, and there are no edges between the subgraphs $\{G[A_1], G[A_2], G[A_3], G[A_4]\}$.

Using the fact that no clique in G' can contain vertices from two distinct sets in $\{A_1, A_2, A_3, A_4\}$, we conclude that G is a positive instance of Fixed-Size 4-CliqueCover with sizes $\{n_1, n_2, n_3, n_4\}$ iff G' is a positive instance of Equal-Size 4-CliqueCover. \square

For the rest of this section we restrict attention to graphs G in the class $\text{GRAPHS}(2^*)$, introduced before Definition 1. A balanced reassembling (G, \mathcal{B}) of such a graph G is relative to a full binary tree \mathcal{B} of height $\log n$ (*i.e.*, with $1 + \log n$ levels) where $n = |V(G)|$.

The nodes in a reassembling tree \mathcal{B} are each a cluster of vertices (Definition 3), a subset of $V(G)$. One of the measures on a node/cluster X in the reassembling (G, \mathcal{B}) is its height (Definition 7), denoted $\text{height}_{G, \mathcal{B}}(X)$. We can extend the measure height to every edge $\overline{vw} \in E(G)$, by defining:

$$\text{height}_{G, \mathcal{B}}(\overline{vw}) := \min \{ \text{height}_{G, \mathcal{B}}(X) \mid X \in \mathcal{B} \text{ and both } v, w \in X \}.$$

In words, the height of \overline{vw} in (G, \mathcal{B}) is the height of the least node/cluster X that includes both endpoints of edge \overline{vw} . Note that, if \mathcal{B} is a full binary tree (the case of a balanced reassembling of $G \in \text{GRAPHS}(2^*)$), then the node/cluster X is at the same distance (or height) from the endpoints (or leaf nodes) v and w .

Another way of understanding $\text{height}_{G, \mathcal{B}}(\overline{vw}) = p$, where $0 \leq p \leq \log n$, is that p is the level number in \mathcal{B} (starting from the bottom, with level 0 being the level of all leaf nodes) at which the two halves of edge \overline{vw} are spliced together, or at which the edge \overline{vw} is re-introduced in the reassembling.

Lemma 17. *If $G \in \text{GRAPHS}(2^*)$ and (G, \mathcal{B}) is a balanced reassembling of G , then:*

$$\beta(G, \mathcal{B}) = 2 \times \sum \{ \text{height}_{G, \mathcal{B}}(\overline{vw}) \mid \overline{vw} \in E(G) \}.$$

Informally, $\beta(G, \mathcal{B})$ is minimized (resp. maximized) when edges are spliced as soon as possible (as late as possible) in the reassembling.

Proof. Straightforward consequence of Definitions 7 and 9. A formal proof can be carried out by induction on $p \geq 1$, where $|V(G)| = n = 2^p$. All details omitted. \square

The proof of the next lemma is interesting in that it combines both algebraic reasoning (formulation of an instance of integer quadratic programming) and combinatorial reasoning (existence of a partition of $V(G)$ into four independent vertex sets of equal size).

Lemma 18. *Let $G \in \text{GRAPHS}(2^*)$ with $|V(G)| = n = 2^p$ for some $p \geq 2$. Let $A_1, A_2, A_3, A_4 \subseteq V(G)$ be four disjoint independent sets of vertices in G , each of size $n/4$. Let (G, \mathcal{B}) be a balanced reassembling and $X_1, X_2, X_3, X_4 \in \mathcal{B}$ be the nodes/vertex-clusters in the tree \mathcal{B} such that $|X_1| = |X_2| = |X_3| = |X_4| = n/4$.³*

Conclusion: *If the β -measure $\beta(G, \mathcal{B})$ is maximized, i.e.,*

$$\beta(G, \mathcal{B}) = \max \{ \beta(G, \mathcal{B}') \mid \mathcal{B}' \text{ is a balanced reassembling tree } \},$$

then $\{X_1, X_2, X_3, X_4\}$ are disjoint independent sets in G , not necessarily the same as $\{A_1, A_2, A_3, A_4\}$, each of size $n/4$.

Proof. Let $V(G) = \{v_1, \dots, v_n\}$. Assume $(X_1 \uplus X_2)$ and $(X_3 \uplus X_4)$ are the two children-nodes/vertex-clusters of the root $V(G)$ in \mathcal{B} . The height in (G, \mathcal{B}) of every vertex cluster in $\{X_1, X_2, X_3, X_4\}$ is $(p-2)$, while the height of both $(X_1 \uplus X_2)$ and $(X_3 \uplus X_4)$ is $(p-1)$, and the height of the root $V(G)$ is of course p .

Lemma 17 gives an alternative definition of $\beta(G, \mathcal{B})$, obtained by summing the heights in (G, \mathcal{B}) of all the edges. This definition is presumed in the formulation of the integer quadratic programming below. The problem of finding a balanced reassembling tree \mathcal{B} which maximizes $\beta(G, \mathcal{B})$ can be translated to an integer quadratic programming, as follows:

$$\begin{aligned} \text{maximize} \quad & \text{(i)} \quad 2p \times \sum_{\overline{v_i v_j} \in E(G)} (x_{i,1}x_{j,3} + x_{i,1}x_{j,4} + x_{i,2}x_{j,3} + x_{i,2}x_{j,4}) & + \\ & \text{(ii)} \quad 2(p-1) \times \sum_{\overline{v_i v_j} \in E(G)} (x_{i,1}x_{j,2} + x_{i,3}x_{j,4}) & + \\ & \text{(iii)} \quad 2(p-2) \times \sum_{\overline{v_i v_j} \in E(G)} (x_{i,1}x_{j,1} + x_{i,2}x_{j,2} + x_{i,3}x_{j,3} + x_{i,4}x_{j,4}) \\ \text{subject to} \quad & \text{(i)} \quad \text{for all } 1 \leq i \leq n \text{ and } 1 \leq k \leq 4, \quad x_{i,k} \in \{0, 1\} \\ & \text{(ii)} \quad \text{for all } 1 \leq k \leq 4, \quad \sum_{1 \leq i \leq n} x_{i,k} = \frac{n}{4} \\ & \text{(iii)} \quad \text{for all } 1 \leq i \leq n, \quad \sum_{1 \leq k \leq 4} x_{i,k} = 1 \end{aligned}$$

The optimization objective is quadratic and has three parts with respective coefficients $2p$, $2(p-1)$, and $2(p-2)$, while the constraints are linear. Every vertex $v_i \in V(G)$ corresponds to four variables $\{x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}\}$, which indicate which set in $\{X_1, X_2, X_3, X_4\}$ contains vertex v_i . Specifically, for every $1 \leq i \leq n$ and $1 \leq k \leq 4$:

$$x_{i,k} = \begin{cases} 1 & \text{if } v_i \in X_k, \\ 0 & \text{if } v_i \notin X_k. \end{cases}$$

To understand the preceding formulation as a $(0, 1)$ quadratic programming, observe that for every edge $\overline{v_i v_j}$:

$$x_{i,k} x_{j,\ell} = \begin{cases} 1 & \text{if } v_i \in X_k \text{ and } v_j \in X_\ell, \\ 0 & \text{if } v_i \notin X_k \text{ or } v_j \notin X_\ell. \end{cases}$$

³ Stated differently, $\{X_1, X_2, X_3, X_4\}$ are the four nodes of \mathcal{B} such that:

$$\text{height}_{G, \mathcal{B}}(X_1) = \text{height}_{G, \mathcal{B}}(X_2) = \text{height}_{G, \mathcal{B}}(X_3) = \text{height}_{G, \mathcal{B}}(X_4) = p-2,$$

where $p = \log n$ with $n = |V(G)|$, i.e., $\{X_1, X_2, X_3, X_4\}$ are the four grandchildren of the root node $V(G)$.

Note that the suggested quadratic system is slightly relaxed in the sense that (as represented by part (iii) with coefficient $2(p-2)$ of the optimization objective) we assume that every edge $\overline{v_i v_j}$ whose endpoints are in the same X_k , i.e., $v_i, v_j \in X_k$ for some $1 \leq k \leq 4$, contributes the maximum possible value, here $2(p-2)$, rather than its exact value to $\beta(G, \mathcal{B})$. The rest of the proof shows that this relaxation does not affect its correctness.

A straightforward re-ordering of terms shows that the optimization objective can be written as follows:

$$\theta := 2p \times \left(\sum_{\overline{v_i v_j} \in E(G), 1 \leq k \leq \ell \leq 4} x_{i,k} x_{j,\ell} \right) - 2\theta_1 - 4\theta_2 \quad \text{where}$$

$$\theta_1 := \sum_{\overline{v_i v_j} \in E(G)} (x_{i,1} x_{j,2} + x_{i,3} x_{j,4}) \quad \text{and}$$

$$\theta_2 := \sum_{\overline{v_i v_j} \in E(G), 1 \leq k \leq 4} x_{i,k} x_{j,k} .$$

The quantity θ_1 counts the number of edges $\overline{v_i v_j}$ satisfying one of two conditions:

- either the endpoints v_i and v_j are in X_1 and X_2 ,
- or the endpoints v_i and v_j are in X_3 and X_4 .

Every edge satisfying one of the two preceding conditions has height $(p-1)$ in (G, \mathcal{B}) . The quantity θ_2 counts the number of edges $\overline{v_i v_j}$ whose endpoints v_i and v_j are in the same X_k , and whose height is therefore $\leq (p-2)$ in (G, \mathcal{B}) . We now observe that:

$$\sum_{\overline{v_i v_j} \in E(G), 1 \leq k \leq \ell \leq 4} x_{i,k} x_{j,\ell} = |E(G)| = m.$$

The optimization objective can now be simplified to read:

$$\theta = 2pm - 2\theta_1 - 4\theta_2 = 2(p-1)m + 2(m - \theta_1 - \theta_2) - 2\theta_2 = 2(p-1)m + \theta' \quad \text{where}$$

$$\theta' := 2(m - \theta_1 - \theta_2) - 2\theta_2 .$$

Hence, maximizing θ is equivalent to maximizing θ' , and the latter is maximized when $(m - \theta_1 - \theta_2)$ is maximized and θ_2 is minimized. But $(m - \theta_1 - \theta_2)$ is the number of edges whose height in (G, \mathcal{B}) is p , i.e., the edges $\overline{v_i v_j}$ such that $v_i \in (X_1 \uplus X_2)$ and $v_j \in (X_3 \uplus X_4)$, while θ_2 is the number of edges $\overline{v_i v_j}$ whose endpoints v_i and v_j are in the same X_k and whose height is $\leq (p-2)$.

We switch to combinatorial reasoning, by invoking the fact that G has four disjoint independent sets, each with $(n/4)$ vertices. There is no need to explicitly solve the integer quadratic programming above. Maximizing θ' means choosing $(X_1 \uplus X_2, X_3 \uplus X_4)$ as a bisection with a maximum cut $|\partial(X_1 \uplus X_2, X_3 \uplus X_4)|$, i.e., choosing $(X_1 \uplus X_2)$ and $(X_3 \uplus X_4)$ as independent sets. And minimizing θ_2 , which is here possible down to $\theta_2 = 0$, means choosing each of X_1, X_2, X_3 , and X_4 , as an independent set. \square

Lemma 19. *Let $G \in \text{GRAPHS}(2^*)$ be a positive instance of Equal-Size 4-CliqueCover, with $|V(G)| = n = 2^p$ for some $p \geq 2$. Let (G, \mathcal{B}) be a balanced reassembling and $\{X_1, X_2, X_3, X_4\} \subseteq \mathcal{B}$ be the nodes/vertex-clusters in the tree \mathcal{B} such that $|X_1| = |X_2| = |X_3| = |X_4| = n/4$.*

Conclusion: *If (G, \mathcal{B}) is a β -optimal balanced reassembling, then $\{X_1, X_2, X_3, X_4\}$ is an Equal-Size 4-CliqueCover of G .*

Proof. Because (G, \mathcal{B}) is a β -optimal balanced reassembling (Definition 9), we have:

$$\beta(G, \mathcal{B}) := \min \{ \beta(G, \mathcal{B}') \mid \mathcal{B}' \text{ is a balanced binary tree over } V(G) \}.$$

We write K_n for the complete graph over $n = |V(G)|$ vertices, and \overline{G} for the complement of G . We thus have $V(\overline{G}) = V(G)$ and $E(\overline{G}) = E(K_n) - E(G)$. Clearly, $G = \overline{\overline{G}}$ and we can write $G = K_n - \overline{G}$. Hence:

$$\beta(G, \mathcal{B}) = \min \{ \beta(K_n - \overline{G}, \mathcal{B}') \mid \mathcal{B}' \text{ is a balanced binary tree over } V(G) \}.$$

The β -measure of a balanced reassembling of K_n , call it M , does *not* depend on the reassembling tree, *i.e.*, for all balanced reassembling trees \mathcal{B}_1 and \mathcal{B}_2 on n vertices, it holds that:⁴

$$\beta(K_n, \mathcal{B}_1) = \beta(K_n, \mathcal{B}_2) = M.$$

Hence, the following equality holds:

$$\beta(G, \mathcal{B}) = M - \max \{ \beta(\overline{G}, \mathcal{B}') \mid \mathcal{B}' \text{ is a balanced binary tree over } V(G) \}.$$

In words, the value of $\beta(G, \mathcal{B})$ is *minimized* when the value of $\beta(\overline{G}, \mathcal{B}')$ is *maximized*. Since (G, \mathcal{B}) is β -optimal, $\beta(\overline{G}, \mathcal{B}')$ is *maximized*.

By hypothesis, G is an instance of Equal-Size 4-CliqueCover, *i.e.*, there are disjoint sets $\{A_1, A_2, A_3, A_4\}$ of vertices in G such that the induced subgraphs in $\{G[A_1], G[A_2], G[A_3], G[A_4]\}$ are each a complete graph with $n/4$ vertices. Hence, the corresponding subgraphs in \overline{G} , namely $\{\overline{G}[A_1], \overline{G}[A_2], \overline{G}[A_3], \overline{G}[A_4]\}$, are each an edgeless graph with $n/4$ vertices. Equivalently, $\{A_1, A_2, A_3, A_4\}$ are disjoint independent sets in \overline{G} , each with $n/4$ vertices. Hence, by Lemma 18, $\{X_1, X_2, X_3, X_4\}$ are disjoint independent sets in \overline{G} , and therefore disjoint cliques in G , each with $n/4$ vertices. \square

Theorem 20. *For the class of simple undirected graphs G , the computation of β -optimal balanced reassemblings (G, \mathcal{B}) is an NP-hard problem.*

Proof. If a deterministic polynomial-time algorithm existed for producing a β -optimal balanced reassembling (G, \mathcal{B}) of an arbitrarily given G , then this algorithm could be used again to decide in deterministic polynomial-time whether a graph in $\text{GRAPHS}(2^*)$ is a positive instance of Equal-Size 4-CliqueCover, by Lemma 19. This would in turn contradict Lemma 16 asserting the NP-completeness of Equal-Size 4-CliqueCover. The desired conclusion follows. \square

5 Related and Future Work

As mentioned in Section 1, graph reassembling can be considered as a special case of a family of graph embedding problems, known as *communication tree embedding* problems. Communication tree embedding is the problem of embedding the vertices of a source graph G into the nodes of a host tree T . In the case where the vertices of G are mapped into the leaves of the host tree, the underlying tree is called a *routing tree* (or call routing tree) and the related problems are referred to as routing tree embedding problems. Graph reassembling is a slight variation of a special case of routing tree embedding where the internal nodes of the host tree have each degree 3, known as *tree layout problem*.

In the context of communication tree embedding, and more specifically tree layout problem, different measures are defined. Corresponding to our α measure in this report is the *edge congestion* measure, which represents the maximum communication traffic on the edges of the host tree T . Seymour and Thomas in [19] show that minimum congestion routing tree problem, referred to as the *minimum carving-width* problem, is solvable in $O(n^4)$ for planar graphs, but is NP-hard in general. The efficiency of the method of Seymour and

⁴We do not need the exact value of M for this proof, it suffices to know it exists, which is an obvious consequence of the fact that every bijection from \mathcal{B}_1 to \mathcal{B}_2 produces an isomorphism (Definition 8) between the balanced reassemblings (K_n, \mathcal{B}_1) and (K_n, \mathcal{B}_2) . It takes some effort to compute M precisely (omitted here): For all balanced reassembling trees \mathcal{B} , if $n = 2^p$, it can be shown that $M = \beta(K_n, \mathcal{B}) = (p-1) \times 2^{2p} + 2^p = ((\log n) - 1) \times n^2 + n$.

Thomas for planar graphs, was improved in [10] and later in [9]. While the result of [19] for minimum carving width of planar graphs can be extended to α -optimal *general* graph reassembling, the status of α -optimal *balanced* graph reassembling for planar graphs is open, which we also conjecture to be NP-hard in contrast to the case of general graph reassembling for planar graphs.

Our β measure in this report corresponds to another measure, known as *tree length*, which is equivalent to the summation of edge congestions in the underlying host tree T . The tree length of a tree embedding represents the average delay (*i.e.*, average dilation) in the source graph and equivalently the average traffic on the edges of the host tree. In [17] it is shown that finding a tree layout with minimum tree length is NP-hard when the source graph is a general graph with no self loops. In the same report also the NP-hardness result is extended to the more general routing tree embedding problem. There are several other measures defined for different variations of communication tree problems; a comprehensive list of these measures can be found in [11, 1, 18].

Future work related to graph reassembling (and specifically related to the balanced case) includes a study of classes of graphs for which α -optimization and/or β -optimization of their reassembling can be carried out in low-degree polynomial times. On the other hand, as indicated in our earlier report [15], the smaller the α and β measures are, the more efficient the execution of programs is, in a domain-specific language (DSL) for the design of flow networks [5, 13, 20, 14]. Beinstock in [6] presents some elementary classes of graphs with small optimal tree congestion as well as an upper bound for the value of optimal tree congestion based on the *tree decomposition* of the graphs. Deciding whether the width of tree decomposition of an arbitrary graph is at most k is NP-complete [3], however the problem is tractable for small and fixed values of k . Bodlaender in [7] surveys a list of graph classes for which the treewidth can be computed in polynomial time. Similarly relating tree width to minimum tree congestion of graphs, [4] characterizes the class of graphs that have α -measure at most 3, but extending these result to different variety of graph reassembling such as balanced case and also study of the characterization of the classes of graphs with higher value of α -measure is open to be investigated in future.

Finally, there is the question of computing approximations of α -optimal and β -optimal balanced graph reassembling, whereby we can turn the NP-hardness of any of the preceding optimizations into polynomially-solvable optimizations. In [16] the problem of finding call routing trees with minimum congestion has been studied and an approximate method for finding a solution within a $O(\log n)$ factor from the optimal solution is suggested. Hence the natural question is if similar methods can be facilitated in order to find approximation of α -optimal and β -optimal for balanced reassembling as well as other variations of graph reassembling.

References

- [1] Carme Àlvarez, Rafel Cases, Josep Díaz, Jordi Petit, and Maria Serna. Communication tree problems. *Theoretical computer science*, 381(1):197–217, 2007. 15
- [2] George E. Andrews. *The Theory of Partitions (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, 1998. 10
- [3] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak -tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. 15
- [4] Rémy Belmonte, Pim van Hof, Marcin Kamiński, Daniël Paulusma, and Dimitrios M Thilikos. Characterizing graphs of small carving-width. In *Combinatorial Optimization and Applications*, pages 360–370. Springer, 2012. 15
- [5] Azer Bestavros and Assaf Kfoury. A Domain-Specific Language for Incremental and Modular Design of Large-Scale Verifiably-Safe Flow Networks. In *Proc. of IFIP Working Conference on Domain-Specific Languages (DSL 2011), EPTCS Volume 66*, pages 24–47, Sept 2011. 15
- [6] Dan Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory, Series B*, 49(1):103–136, 1990. 15
- [7] Hans L Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2):1, 1994. 15
- [8] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976. 2, 3
- [9] Qian-Ping Gu and Hisao Tamaki. Optimal branch-decomposition of planar graphs in $o(n^3)$ time. *ACM Transactions on Algorithms (TALG)*, 4(3):30, 2008. 15

- [10] Illya V Hicks. Planar branch decompositions ii: The cycle method. *INFORMS Journal on Computing*, 17(4):413–421, 2005. 15
- [11] Jordi Petit i Silvestre. *Layout problems*. PhD thesis, Ph. D. thesis, Universitat Politècnica de Catalunya, Barcelona, 25 May, 2001. 15
- [12] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972. 2, 4
- [13] Assaf Kfoury. The Syntax and Semantics of a Domain-Specific Language for Flow-Network Design. *Science of Computer Programming*, 93(Part A):19–38, November 2014. 15
- [14] Assaf Kfoury and Saber Mirzaei. A Different Approach to the Design and Analysis of Network Algorithms. Technical Report BUCS-TR-2012-019, CS Dept, Boston Univ, 2013. 15
- [15] Assaf Kfoury and Saber Mirzaei. Efficient reassembling of graphs, part 1: The linear case. *Accepted for publication in the Journal of Combinatorial Optimization*, 2016. 1, 2, 3, 4, 15
- [16] Samir Khuller, Balaji Raghavachari, and Neal Young. Designing multi-commodity flow trees. *Information Processing Letters*, 50(1):49–55, 1994. 15
- [17] Saber Mirzaei. Minimum average delay of routing trees. *arXiv preprint arXiv:1601.02697*, 2016. 15
- [18] Jordi Petit. Addenda to the Survey of Layout Problems. *Bulletin of the EATCS*, (105):177–201, October 2011. 15
- [19] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994. 14, 15
- [20] Nate Soule, Azer Bestavros, Assaf Kfoury, and Andrei Lapets. Safe Compositional Equation-based Modeling of Constrained Flow Networks. In *Proc. of 4th Int'l Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Zürich, September 2011. 15